GWD-I
Grid Benchmarking

R.F. Van der Wijngaart
Computer Sciences Corporation
M.A. Frumkin
NASA Ames Research Center
January 2003

Computationally Intensive Grid Benchmarks

Abstract

This documents provides a paper-and-pencil specification of a suite of four families of benchmarks for measuring the performance of grids when executing computationally intensive applications. The Computationally Intensive Grid Benchmarks (CIGB), developed at NASA's Advanced Supercomputing division and originally called the NAS Grid Benchmarks, are based on the NAS Parallel Benchmarks (NPB). The CIGB are presented as data flow graphs encapsulating an instance of a slightly modified NPB task in each graph node, which communicates with other nodes by sending/receiving initialization data. Like NPB, CIGB specify several different classes (problem sizes). In this document we describe classes S, W, A, and B, and provide verification values for each. The implementor has the freedom to choose any language, grid environment, security model, fault tolerance/error correction mechanism, etc., as long as the resulting implementation pass a verification test and reports the turnaround time of the benchmark.

Contents

1 Introduction

The Computationally Intensive Grid Benchmarks (CIGB) consist of four families of synthetic applications that are dominated by computation. They test the ability of grid environments to execute distributed processes. Three of the four families require the constituent processes to exchange data. The benchmarks are called computationally intensive because they perform a fair number of arithmetic operations for each word exchanged between processes. CIGB were developed at NASA's Advanced Supercomputing division (NAS), and were originally called the NAS Grid Benchmarks (NGB) [3, 4]. They are based on the NAS Parallel Benchmarks (NPB), whose specification is provided in a report available from NAS [1]. This report describes only the modifications to the NPB required to implement CIGB.

The pencil-and-paper specification provided here serves as a uniform tool for testing functionality and efficiency of grid environments. Users are free to implement CIGB as they see fit, provided they observe the same—fairly loose—rules laid down in the NPB [1] report. A correct reference implementation is available from NAS under the name GridNPB as part of the NPB software suite [5]. CIGB specifies the problem to be solved by each task and the data to be communicated between the tasks. However, it does not specify how to implement/select the following: authentication, security, fault tolerance, scheduling, grid environment, mapping of CIGB tasks onto the computational grid. A CIGB result consists of a correctly reproduced set of verification values, plus the turnaround time. The grid configuration, specifically the aggregate resources (disk space, CPU time, memory, network bandwidth) used to complete the benchmarks, is currently considered too poorly defined to have utility outside the benchmarker's own organization. A CIGB implementor may provide a more detailed report on the performance of grid components, including time to communicate data between any two benchmark tasks, and wall clock time for each task. But since CIGB does not specify how many or which resources to employ, the detailed report is considered informative in nature.

2 CIGB Data Flow Graphs

An instance of CIGB comprises a collection of slightly modified NPB problems, each defined on a fixed, rectilinear discretization mesh. Each NPB problem used (BT, SP, LU, MG, or FT) is specified by class (mesh size, number of iterations), source(s) of input data, and consumer(s) of solution values. Hence, an instance of CIGB can be specified by a *Data Flow Graph* (DFG), see Figures 1–4. The DFG consists of nodes connected by directed arcs. It is constructed such that there is a directed path from any node to the sink node of the graph (indicated by *Report*). This is necessary to ensure that any failing node will be detected.

2.1 Corrections to NPB

It has been observed by a number of researchers that the officially released MPI implementation of NPB [2] and the paper-and-pencil specification [1] contain some inaccuracies and minor errors. Two of these cause problems when implementing CIGB. We list them here, plus their corrections.

- The MPI implementation of FT does not scale the reverse Discrete Fourier Transform with the size of the mesh as it should according to [1]. This causes the norm of the solution field after each invocation of FT within CIGB to jump by a factor of $n_1 \times n_2 \times n_3$, where n_i is the number of mesh points in coordinate direction i. Especially for the larger problem sizes the jump becomes too large, so we always divide the NPB FT result by $n_1 \times n_2 \times n_3$ before transferring the (real part of) the solution to a successor DFG node, but after computing checksums in case the node performs a verification test.

- Initialization of the flow field in SP, BT, and LU is supposed to employ transfinite interpolation of the exact solution on the boundaries of the discretization mesh. However, in neither the NPB specification [1] nor its MPI implementation does the initialization correspond to any reasonable interpolation. We remedy this by employing tri-linear (*not* transfinite) interpolation (see Section 2.2 below) to compute the initial flow field for SP, BT, and LU whenever they are immediate successors of the Launch node. In this process only the values of the dependent variables at the eight corners of the cubical grid are used.

2.2 Pre- and Post-processing

If multiple input fields are supplied to a graph node (multiple input arcs), or if the type of input data does not match the size of the receiving mesh or the data type on which the graph node representing the NPB task operates, some data pre-processing is required. Similarly, if the output field of a graph node does not match the data type of the consumer of that data, some post-processing is required. The distinction between pre-processing and post-processing is fairly arbitrary.

Pre-processing

All NPB problems used within CIGB are defined on the three-dimensional unit cube. However, even within the same problem class (S, W, A, or B) there are different mesh sizes for the different benchmark tasks. Discretization points of meshes of different size generally do not coincide. In order to use the output from one or more NPB tasks as input for another, we interpolate the data tri-linearly as needed, and subsequently compute arithmetic averages at each mesh point in case of multiple input arcs. Let a variable u be defined at the grid points of a mesh of extent $(1{:}nx_1, 1{:}ny_1, 1{:}nz_1)$, and let v be the interpolant of the same variable at the grid points of a mesh of extent $(1{:}nx_2, 1{:}ny_2, 1{:}nz_2)$. The value of v at point (i, j, k) is calculated as follows.

$$\begin{aligned} v(i,j,k) = \quad & \gamma \quad [\quad \beta \quad (\alpha u(i_h, j_h, k_h) + (1-\alpha) u(i_l, j_h, k_h)) + \\ & \qquad (1-\beta) \, (\alpha u(i_h, j_l, k_h) + (1-\alpha) u(i_l, j_l, k_h))] \\ & + \\ & (1-\gamma) [\quad \beta \quad (\alpha u(i_h, j_h, k_l) + (1-\alpha) u(i_l, j_h, k_l)) + \\ & \qquad (1-\beta) \, (\alpha u(i_h, j_l, k_l) + (1-\alpha) u(i_l, j_l, k_l))] \end{aligned} \tag{1}$$

where

$$\begin{array}{lll} dx = 1/(nx_2 - 1) & dy = 1/(ny_2 - 1) & dz = 1/(nz_2 - 1) \\ x = (i-1)dx(nx_1 - 1) & y = (j-1)dy(ny_1 - 1) & z = (k-1)dz(nz_1 - 1) \\ i_h = \min(\lfloor x + 2 \rfloor, nx_1) & j_h = \min(\lfloor y + 2 \rfloor, ny_1) & k_h = \min(\lfloor z + 2 \rfloor, nz_1) \\ i_l = i_h - 1 & j_l = j_h - 1 & k_l = k_h - 1 \\ \alpha = x - i_l & \beta = y - j_l & \gamma = z - k_l \end{array} \tag{2}$$

If u and v are vector fields the interpolation is performed in a componentwise fashion.

When FT receives input data, part or all always originates from an MG NPB task. FT operates on double precision complex scalar data, whereas MG manipulates double precision real scalar data. The mapping from real output data to complex input data is as follows. The real part of the complex input data is the same as the arithmetic average at each mesh point of the MG real output value (interpolated to the correct mesh) and the real part of the FT complex output value (if present). The imaginary part is a scrambled version of the real part. For mesh point (i, j, k) of the FT mesh (the origin of the mesh is point $(1, 1, 1)$) the imaginary part of the initial solution u is:

$$Im(u) = (((i + j + k) \bmod 3) - 1) * Re(u)\,. \tag{3}$$

Post-processing

When BT or LU output data to be sent to an MG node–which operates on data that consists of a single double precision word per mesh point–they compute the local speed of sound a for each mesh point. The solution at a point is defined by the vector $\mathbf{u}$, with five components (see [1]). The speed of sound is defined by:

$$a = \left(0.56(u_5 - \frac{1}{2u_1}(u_2^2 + u_3^2 + u_4^2))/u_1\right)^{1/2}. \tag{4}$$

When MG outputs data to be sent to an FT node, it transfers the solution on the entire mesh (a single double precision word per point), except the solution values on the periodic boundaries. When FT outputs data to be sent to an FT node, as happens in the VP benchmark, only the real part of the complex solution value on the entire mesh is transferred.
Table 1 lists the types of data exchanged between NPB tasks within CIGB, as defined by the receiving task.

Table 1: Types of field data (double precision real) exchanged by NPB tasks

Arc tail \ Arc head	BT	SP	LU	MG	FT
BT	5-vector	5-vector	5-vector	scalar	–
SP	5-vector	–	5-vector	–	–
LU	5-vector	–	–	scalar	–
MG	–	–	–	–	scalar
FT	–	–	–	–	real part of complex scalar

2.3 DFG Node

Each node (except *Launch* and *Report*) represents a single computational task, which consists of solving one of the NPB problems. It has a set of input and output

arcs, and a compute module. If a node is connected to the source node (indicated by *Launch* in Figures 1–4), it receives control directives to initiate the computation. Otherwise it receives input data from other nodes through its input arc(s), to be used to calculate or set initial conditions. If the node is not connected to the sink node (indicated by *Report* in Figures 1–4), it sends the computed solution along all output arcs. The implementor is free to attach to a node any additional attributes, such as information on the computational resources required for performing its functions, which can be used by a scheduler. The sink node collects verification statuses of any nodes connected to it. Note: the structuring of the CIGB as DFGs serves only as a conceptual description. The implementor is free to modify the granularity of the benchmark by merging or splitting nodes.

2.4 DFG Arc

An arc connects tail and head nodes and represents transmission of data from the tail to the head. The implementor is free to attach to the arc any additional attributes, such as information on the communication volume and frequency, which can be used by a scheduler. Data to be exchanged between DFG nodes may not be precomputed or cached, but must be created anew for each benchmark run. Dashed arcs in Figures 1–4 connect the nodes *Launch* and *Report* to the rest of the graph. They carry no computational data, but are required for control and timing. CIGB does not prescribe the mechanism for transferring data, nor does it prescribe the representation of data to be exhanged between DFG nodes. This has some implications for the verification tests performed by the nodes connected to the Report node, as descibed below.

2.5 Four CIGB Problems

CIGB consists of four families of problems, named Embarrassingly Distributed (ED), Helical Chain (HC), Visualization Pipeline (VP), and Mixed Bag (MB). They are described in detail in the following sections.
Scaleup of CIGB problems from class S to larger problem sizes is accomplished by increasing the number of graph nodes, as described in Section 2.6, as well as the size of the NPB problem in each node. Specifically, for CIGB of class X we employ only NPB problems of class X.
Other than ED, each CIGB problem involves transfer of solution values between DFG nodes. This is a potential source of numerical error, because nodes may execute on different architectures, with different data representations and/or arithmetic. The original NPB verification tests are fairly tight, to avoid the possibility of validating an erroneous solution. Since CIGB uses the same rather strict error tolerances, it is possible that correctly computed solutions fail the verification test. Specifically, if error norms of the final CIGB solution(s) have a small absolute magnitude, the build-up of errors due to differing arithmetic or to data conversions may exceed the verification threshold. For example, if the magnitude of an actual error norm is 10^{-7} and data conversion causes differences in the last bit of a double precision number with a 48-bit mantissa, an error tolerance of 10^{-7} may cause a correctly computed

solution to fail the verification test. To avoid this problem, which manifests itself for class S of the CIGB, we cut the size of the time step of SP, BT, and LU for that class in half (except for ED, which does not suffer from data conversions, because it employs the original, non-communicating SP NPB problem). This slows down convergence sufficiently that final error norms are well above the threshold value for triggering false verification failures.

2.5.1 Embarrassingly Distributed (ED)

ED represents the class of grid applications called parameter studies, which constitute multiple independent runs of the same program, but with different input parameters.

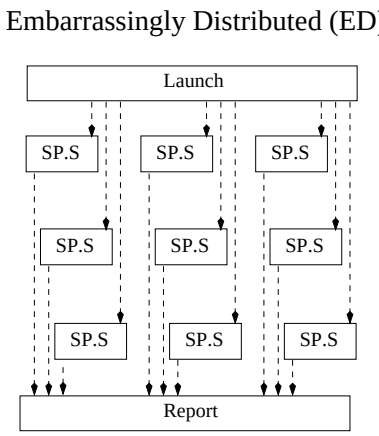

Figure 1: Data flow graph of CIGB problem ED, class S (sample size). Dashed arrows signify control flow.

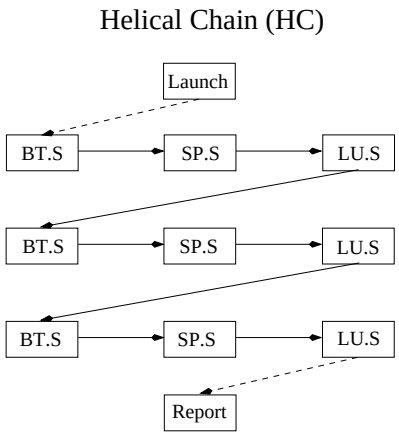

Figure 2: Data flow graph of CIGB problem HC, class S (sample size). Solid arrows signify data and control flow. Dashed arrows signify control flow only.

We select SP, the core of an important class of flow solvers, as the basis for this benchmark. There is no communication between any of the instantiations of SP, as indicated in Figure 1 depicting class S (sample size) of the benchmark. If we number the nodes of the ED graph consecutively, starting from zero, then the parameter study is defined by varying the initialization constant $C_{1,1}$, as defined in [1], as follows: $C_{1,1} = 2(1 + i * 0.01)$, where i is the node number. Note that the initialization of the flow field in the interior of the mesh takes place through tri-linear interpolation of the flow variables at the eight corner points of the mesh, see Section 2.1. No other changes are made to the NPB problem defining SP.

2.5.2 Helical Chain (HC)

HC represents chains of repeating processes. We select BT, SP, and LU to make up the successive nodes in the chain, and connect this triplet into a linear chain, as shown in Figure 2. Initialization of the computation takes place in the regular NPB style for the first BT node in the graph (but see Section 2.1 for the correction we need to apply). Subsequent nodes use the final computed solution of their predecessor node to initialize. For problem classes S, A, and B no interpolation is required, because the mesh sizes for SP, BT, and LU are identical. However, for class W they differ, so an interpolation is required in a pre-processing step (see Section 2.2).

2.5.3 Visualization Pipeline (VP)

VP represents chains of compound processes, like those encountered when visualizing flow solutions as the simulation progresses. It comprises the three NPB problems BT, MG, and FT, which fulfill the role of flow solver, post processor, and visualization module, respectively. This triplet is linked into a logically pipelined process, where subsequent flow solutions can be computed while postprocessing and visualization of previous solutions is still in progress. This process is illustrated in Figure 3.

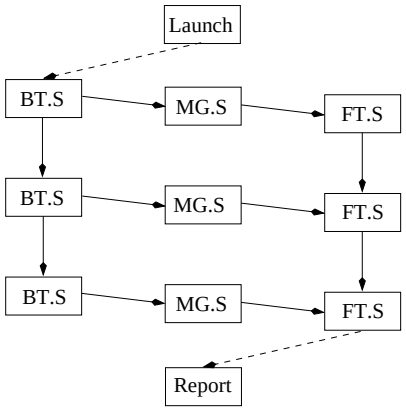

Figure 3: Data flow graph of CIGB problem VP, class S (sample size). Solid arrows signify data and control flow. Dashed arrows signify control flow only.

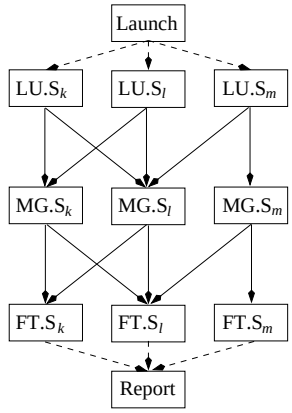

Figure 4: Data flow graph of CIGB problem MB, class S (sample size). Solid arrows signify data and control flow. Dashed arrows signify control flow only. Subscripts indicate different (relative) numbers of iterations.

Data exchange between NPB nodes in VP is as follows. Each BT node transfers its entire solution to its BT successor node in the pipeline (if present), with no

interpolation necessary. Initialization of the computation takes place in the regular NPB style for the first BT node in the graph (but see Section 2.1 for the correction we need to apply). From BT's vector output field we can compute the scalar field of sound speeds, consisting of one double precision number for each point in the BT mesh. This solution is used to initialize the MG successor node, but only in the interior of the mesh. Hence, the BT sound speed field is interpolated onto the interior of the MG mesh. This interior covers the unit cube, and thus coincides with the BT mesh in physical space. The boundary values for MG are set by using explicit periodic boundary conditions on all six faces of the cubic mesh, which means copying initialization values at discretization points one cell away from the mesh boundary to the corresponding boundary location on the other side of the mesh (this copying process is identical to that in the original NPB MG problem specification within each iteration). Similarly, upon completion MG transfers the interior values of its computed solution to the FT node. However, for each FT node other than the first in the visualization pipeline there is also an FT solution that is used to initialize the successor FT node. As mentioned in Section 2.2, FT uses for its initialization an array of double precision real values, consisting of the arithmetic average of the real part of the previous FT solution (if present) and the interior solution of the MG node, interpolated onto the whole FT mesh.

2.5.4 Mixed Bag (MB)

MB is similar to VP. It again involves the sequence of flow computation, post-processing, and visualization, but now the emphasis is on introducing asymmetry. Different amounts of data are transferred between different tasks, and some tasks require more work than others, making it hard for a grid scheduler to map DFG tasks to grid resources efficiently. The MB DFG specifies that several flow computations can start simultaneously (the first horizontal layer), but the next layer implies some synchronization when computed solutions from multiple instantiations of LU are used as input for the MG nodes, see Figure 4. The same communication/synchronization pattern is used for transfer of data between all graph layers. Whenever a node has multiple input arcs, the arithmetic average of the (interpolated) solutions at each grid point is computed and passed to the successor node. As in the case of VP, the interior of the MG mesh is initialized with the sound speed field from the entire mesh of its NPB predecessor node (LU in this case), and MG also transfers solution values on the interior of its mesh to the entire mesh of FT successor nodes. Also, as in the case of VP, the double precision complex initial field of FT is constructed from double precision real data using Equation 3. An additional complexity of MB is that nodes in different columns of the DFG perform different (relative) numbers of time steps/iterations, indicated by the subscripts k, l, and m in Figure 4. This creates a potential for load imbalance that must be handled by allocating different amounts of resources to computational nodes in the same layer of the graph. The mechanism for determining the relative number of iterations for the graph nodes is as follows. Let the number of iterations for a node in the leftmost column in graph layer d be N_d^0, which is determined by dividing the number of iterations of the original NPB by the *depth* of the DFG, with a minimum

of one, as described in Section 2.6. Also, let the column index of a graph node be c, starting with zero for the leftmost column, and increasing with unit steps when moving to the right[1]. Then the number of iterations for any node is defined by $N_d^c = \max(1, \lfloor N_d^0(1 - \frac{c}{2(c+1)}) \rfloor)$. For class S the numbers of iterations executed by LU nodes of increasing column index are 16, 12, and 10, by MG nodes 1, 1, and 1, and by FT nodes 2, 1, and 1, respectively. Initialization of the computations in the first graph layer is the same as in the regular NPB (but see Section 2.1 for the correction we need to apply).

2.6 Scaling Rationale and Verification

The purpose of CIGB is to gauge the capabilities of grids when executing distributed tasks that are dominated by computation, that is, that perform a fair number of arithmetic operations for each word exchanged between processes. It is expected that as time progresses, such grids will become more powerful, both in terms of single system performance, as well as in terms of the number of accessible systems. The creation of larger CIGB problem sizes (classes) is meant to capture this expected development. Hence, successive CIGB classes will involve more computational work, as well as more graph nodes, in general. The rationale for selecting the parameters that determine these is as follows.

The NPB classes already capture growth in problem size. CIGB makes use of this by employing for each class a data set (mesh or array) of the same size as the corresponding NPB class. Furthermore, we accept the premise that a CIGB instance of a certain class should run approximately as long as an instance of NPB of the same class if we disregard communication times between graph nodes, pre- and post-processing, and exploitation of intra-node parallelism. In other words, we associate with each graph node a weight equal to the amount of computational work, and make sure that the critical path has approximately the same aggregate weight as an NPB instance of that class. Other than ED, each CIGB problem will have several different NPB problems on its critical path. The goal of keeping the summed weights of the nodes on the critical path the same as that of a single NPB problem may be reached by setting the number of iterations or time steps within each CIGB NPB problem equal to that of the original NPB problem, divided by the number of nodes on the critical path, rounded down, with a minimum of one. Consequently, the computational work on the CIGB critical path will be no more than that of the most computationally intensive NPB problem of the same class.

Note: we define the *width* of each CIGB DFG as the maximum number of nodes in the DFG that can be executed concurrently, and the *depth* as the length of the critical path of the DFG, ignoring the pipeline fill or drain nodes in VP. Using these definitions, the total size of each CIGB DFG, not including the Report and Launch nodes, is the product of depth and width. For convenience we use the depth instead of critical path length for scaling the number of iterations or time steps within the DFGs. DFG width has no influence on turnaround time—save pipeline fill/drain

[1]For class S the FT node with column index zero has two input arcs, that with column index one has three input arcs, and that with column index two has one input arc.

time—if the CIGB implementor fully exploits inter-node parallelism.
If we map consecutive classes S, W, A, B, ... to consecutive integers, starting with one, we specify the depth D of CIGB class i as follows.
$D^i_{ED} = 1, D^i_{HC} = 9\max(1, i-2), D^i_{VP} = 3\max(1, i-2), D^i_{MB} = \max(3, i)$.
The corresponding width W is defined as follows.
$W^i_{ED} = 9 * 2^{\max(0,i-3)}, W^i_{HC} = 1, W^i_{VP} = 3, W^i_{MB} = \max(3, i)$.
We list the numerical values of the total numbers of nodes for benchmark classes S, W, and A through E, respectively, in Table 2. While it is clear which NPB problems to select for the different graph layers of ED, HC, and VP, it is not obvious for MB. We adopt the following strategy. The last graph layer (connected to the Report node) is assumed to have depth zero. Depth increases by one for each higher layer. Layers at depths 0 and 1 always employ FT and MG, respectively. Layers at larger depth are assigned NPB problems LU, BT, and SP, respectively, in a cyclical fashion.

Table 2: Width × depth of CIGB graphs

	Class						
Name	S	W	A	B	C	D	E
ED	9 × 1	9 × 1	9 × 1	18 × 1	36 × 1	72 × 1	144 × 1
HC	1 × 9	1 × 9	1 × 9	1 × 18	1 × 27	1 × 36	1 × 45
VP	3 × 3	3 × 3	3 × 3	3 × 6	3 × 9	3 × 12	3 × 15
MB	3 × 3	3 × 3	3 × 3	4 × 4	5 × 5	6 × 6	7 × 7

Verification values for CIGB classes S, W, A, and B are presented in the Appendix. For ED these values are required for each node in the graph, for HC and VP only for the last node, and for MB for all nodes in the last layer of the graph. Verification procedures for ED and HC are the same as those for the original NPB, although the actual values differ, of course. For VP and MB verification values and procedures must be defined for FT nodes, since these are connected to the report node. In the original NPB each step in the reverse FFT computes a double precision complex checksum and compares it with a verification value. The reason for this is that the reverse FFT steps are independent, and verifying only the last is not sufficient. In CIGB the number of verification values is potentially significantly larger than in NPB. To keep that number within reason, we verify not every reverse FFT step individually, but compute the arithmetic average of the checksums over all the reverse FFT steps within each FT task and compare that to a verification value.

3 Security Considerations

The paper-and-pencil specification of CIGB does not stipulate any requirements regarding security.

4 Author Contact Information

Rob F. Van der Wijngaart, Michael A. Frumkin
Mail Stop T27A-1
NASA Ames Research Center
Moffett Field, CA 94035-1000
{wijngaar,frumkin}@nas.nasa.gov

5 Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the GGF Secretariat. The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director (see contacts information at GGF website).

6 Full Copyright Notice

7 Appendix: Verification values

The CIGB require that the correct solutions be computed. This is ensured by demanding that verification values for solution and error norms of all nodes that are directly connected to the Report node in the DFG are computed. In this section we provide these values. Error tolerances and methods for computing norms for CIGB are the same as for NPB.

7.1 Embarrassingly Distributed

Verification values for ED, class S, computed by each SP node

node #	solution norm	error norm
0	0.8676543011741d-04	0.8719352691803d-07
	0.6033493408310d-04	0.5988808458188d-07
	0.6278114032528d-04	0.6270092156558d-07
	0.6603179902638d-04	0.6551568383242d-07
	0.9122028866543d-04	0.8553969317102d-07
1	0.8601620225691d-04	0.8643202500288d-07
	0.5988773196543d-04	0.5943418706437d-07
	0.6157026954073d-04	0.6155109448566d-07
	0.6457044508652d-04	0.6413646078612d-07
	0.8687428401194d-04	0.8152493003792d-07
2	0.8528517607623d-04	0.8568916795314d-07
	0.5944509868286d-04	0.5898555351009d-07
	0.6043095623721d-04	0.6046719704763d-07
	0.6319041663855d-04	0.6283245382631d-07
	0.8273663135464d-04	0.7770753603531d-07
3	0.8457166533017d-04	0.8496424291102d-07
	0.5900696189910d-04	0.5854206620123d-07
	0.5935880462626d-04	0.5944517278358d-07
	0.6188715115064d-04	0.6159941928907d-07
	0.7879773267341d-04	0.7407863788365d-07
4	0.8387501686770d-04	0.8425657226531d-07
	0.5857325969939d-04	0.5810362059449d-07
	0.5834963218012d-04	0.5848116700380d-07
	0.6065629301996d-04	0.6043330938920d-07
	0.7504851869600d-04	0.7062984804457d-07
5	0.8319460928597d-04	0.8356551209019d-07
	0.5814393866091d-04	0.5767012327603d-07
	0.5739946515836d-04	0.5757152229935d-07
	0.5949368685757d-04	0.5933026612155d-07
	0.7148042299123d-04	0.6735324051262d-07

node #	solution norm	error norm
6	0.8252985024299d-04	0.8289044920091d-07
	0.5771895249772d-04	0.5724149042257d-07
	0.5650453308785d-04	0.5671277285472d-07
	0.5839537229112d-04	0.5828661632209d-07
	0.6808535665169d-04	0.6424132668705d-07
7	0.8188017551638d-04	0.8223080020494d-07
	0.5729826099434d-04	0.5681764661514d-07
	0.5566126383428d-04	0.5590163933420d-07
	0.5735757816226d-04	0.5729886620119d-07
	0.6485568447004d-04	0.6128703228573d-07
8	0.8124504686581d-04	0.8158600910848d-07
	0.5688182884348d-04	0.5639852350520d-07
	0.5486627790428d-04	0.5513502305850d-07
	0.5637671694675d-04	0.5636369592773d-07
	0.6178420280687d-04	0.5848367565086d-07

Verification values for ED, class W

node #	solution norm	error norm
0	0.1745133059397d-04	0.6955341579879d-06
	0.1194347497651d-04	0.4732433767510d-06
	0.1271275032480d-04	0.5071720177863d-06
	0.1150480909799d-04	0.4600674574080d-06
	0.1305114022206d-04	0.5381322068507d-06
1	0.1745560352290d-04	0.6957392189784d-06
	0.1193739401471d-04	0.4730069614536d-06
	0.1269942115908d-04	0.5066479195335d-06
	0.1149066787325d-04	0.4595086246319d-06
	0.1301167555462d-04	0.5365356761184d-06
2	0.1745986923555d-04	0.6959438483249d-06
	0.1193141412701d-04	0.4727745781123d-06
	0.1268621099471d-04	0.5061284384576d-06
	0.1147665934807d-04	0.4589549875939d-06
	0.1297237680057d-04	0.5349455062024d-06
3	0.1746412767819d-04	0.6961480441358d-06
	0.1192553395452d-04	0.4725461723920d-06
	0.1267311889347d-04	0.5056135396850d-06
	0.1146278217756d-04	0.4584064944105d-06
	0.1293324425385d-04	0.5333617144459d-06
4	0.1746837879467d-04	0.6963518044347d-06
	0.1191975215752d-04	0.4723216907465d-06
	0.1266014391456d-04	0.5051031881884d-06
	0.1144903502624d-04	0.4578630935544d-06
	0.1289427815335d-04	0.5317843158727d-06

node #	solution norm	error norm
	0.1747262252970d-04	0.6965551273059d-06
	0.1191406741612d-04	0.4721010804231d-06
5	0.1264728511460d-04	0.5045973487811d-06
	0.1143541657009d-04	0.4573247339124d-06
	0.1285547868348d-04	0.5302133232448d-06
	0.1747685882701d-04	0.6967580108191d-06
	0.1190847842890d-04	0.4718842894206d-06
6	0.1263454154958d-04	0.5040959861974d-06
	0.1142192549392d-04	0.4567913647034d-06
	0.1281684597929d-04	0.5286487472472d-06
	0.1748108763020d-04	0.6969604530568d-06
	0.1190298391404d-04	0.4716712665253d-06
7	0.1262191227311d-04	0.5035990650298d-06
	0.1140856049497d-04	0.4562629356041d-06
	0.1277838012802d-04	0.5270905965546d-06
	0.1748530888245d-04	0.6971624520995d-06
	0.1189758260757d-04	0.4714619612488d-06
8	0.1260939633998d-04	0.5031065498690d-06
	0.1139532028062d-04	0.4557393966743d-06
	0.1274008117128d-04	0.5255388779393d-06

Verification values for ED, class A

node #	solution norm	error norm
	0.1662388872593d-03	0.6779616046334d-06
	0.1120336284839d-03	0.4612029937003d-06
0	0.1155494269554d-03	0.4944689998311d-06
	0.1144591065562d-03	0.4485901415827d-06
	0.1164120101231d-03	0.5246454435912d-06
	0.1643876342654d-03	0.6781623235228d-06
	0.1107586934055d-03	0.4609720478128d-06
1	0.1139204792057d-03	0.4939569076487d-06
	0.1126757070066d-03	0.4480440569950d-06
	0.1116344021422d-03	0.5230883044555d-06
	0.1625918474544d-03	0.6783626225632d-06
	0.1095197898803d-03	0.4607450429708d-06
2	0.1123587585230d-03	0.4934493367655d-06
	0.1109676929314d-03	0.4475030647124d-06
	0.1071044513952d-03	0.5215374339158d-06
	0.1608490693651d-03	0.6785624992312d-06
	0.1083153581826d-03	0.4605219257276d-06
3	0.1108604672866d-03	0.4929462520671d-06
	0.1093308651249d-03	0.4469671125838d-06
	0.1028086486843d-03	0.5199928417464d-06

node #	solution norm	error norm
	0.1591569849646d-03	0.6787619510589d-06
	0.1071439334321d-03	0.4603026434516d-06
4	0.1094220422343d-03	0.4924476183695d-06
	0.1077612746219d-03	0.4464361489565d-06
	0.9873431034114d-04	0.5184545362024d-06
	0.6789609756324d-06	0.6789609756324d-06
	0.4600871443050d-06	0.4600871443050d-06
5	0.4919534004407d-06	0.4919534004407d-06
	0.4459101226618d-06	0.4459101226618d-06
	0.5169225240005d-06	0.5169225240005d-06
	0.1559163057636d-03	0.6791595706091d-06
	0.1048946693312d-03	0.4598753772286d-06
6	0.1067116281816d-03	0.4914635630149d-06
	0.1048091731788d-03	0.4453889830283d-06
	0.9120310319412d-04	0.5153968104229d-06
	0.1543637155168d-03	0.6793577337061d-06
	0.1038143074591d-03	0.4596672919413d-06
7	0.1054335631917d-03	0.4909780708165d-06
	0.1034198910287d-03	0.4448726798508d-06
	0.8772452336450d-04	0.5138773993154d-06
	0.1528538118743d-03	0.6795554626864d-06
	0.1027618953918d-03	0.4594628389051d-06
8	0.1042031878741d-03	0.4904968885790d-06
	0.1020842773640d-03	0.4443611634404d-06
	0.8442389321125d-04	0.5123642931624d-06

Verification values for ED, class B

node #	solution norm	error norm
	0.2888397222677d-02	0.8570799745568d-04
	0.1986283266871d-02	0.6084340831742d-04
0	0.2056836823212d-02	0.6168313995579d-04
	0.1919068081146d-02	0.5503556021184d-04
	0.1760373599697d-02	0.5371992957896d-04
	0.2874533169716d-02	0.8571710411892d-04
	0.1976782088870d-02	0.6081518097910d-04
1	0.2043753689422d-02	0.6163071857748d-04
	0.1904596881575d-02	0.5498088125148d-04
	0.1727497444863d-02	0.5356554133133d-04
	0.2861164686458d-02	0.8572624719782d-04
	0.1967613387202d-02	0.6078737549820d-04
2	0.2031202225569d-02	0.6157872751497d-04
	0.1890716228756d-02	0.5492668080331d-04
	0.1696473877754d-02	0.5341176861583d-04

node #	solution norm	error norm
	0.2848268089926d-02	0.8573542557726d-04
	0.1958760852630d-02	0.6075998643316d-04
3	0.2019153080765d-02	0.6152716432492d-04
	0.1877394102056d-02	0.5487295478700d-04
	0.1667189904025d-02	0.5325861421782d-04
	0.2835821109129d-02	0.8574463815689d-04
	0.1950209192699d-02	0.6073300841469d-04
4	0.2007578750517d-02	0.6147602651296d-04
	0.1864600427045d-02	0.5481969912654d-04
	0.1639539059997d-02	0.5310608065702d-04
	0.2823802783533d-02	0.8575388385133d-04
	0.1941944055995d-02	0.6070643614528d-04
5	0.1996453452999d-02	0.6142531153808d-04
	0.1852306952467d-02	0.5476690975254d-04
	0.1613421072032d-02	0.5295417020011d-04
	0.2812193372356d-02	0.8576316159037d-04
	0.1933951963416d-02	0.6068026439855d-04
6	0.1985753017866d-02	0.6137501681678d-04
	0.1840487137361d-02	0.5471458260442d-04
	0.1588741534434d-02	0.5280288487242d-04
	0.2800974268452d-02	0.8577247031912d-04
	0.1926220244872d-02	0.6065448801860d-04
7	0.1975454779118d-02	0.6132513972700d-04
	0.1829116044815d-02	0.5466271363248d-04
	0.1565411598655d-02	0.5265222646928d-04
	0.2790127921639d-02	0.8578180899813d-04
	0.1918736982430d-02	0.6062910191940d-04
8	0.1965537476502d-02	0.6127567761178d-04
	0.1818170239707d-02	0.5461129879968d-04
	0.1543347683346d-02	0.5250219656663d-04
	0.2779637767167d-02	0.8579117660351d-04
	0.1911490955130d-02	0.6060410108407d-04
9	0.1955981162215d-02	0.6122662778274d-04
	0.1807627694343d-02	0.5456033408342d-04
	0.1522471192103d-02	0.5235279653128d-04
	0.2769488159399d-02	0.8580057212696d-04
	0.1904471592179d-02	0.6057948056416d-04
10	0.1946767114218d-02	0.6117798752330d-04
	0.1797467699474d-02	0.5450981547709d-04
	0.1502708249421d-02	0.5220402753050d-04
	0.2759664309264d-02	0.8580999457585d-04
	0.1897668927284d-02	0.6055523547899d-04
11	0.1937877755297d-02	0.6112975409179d-04
	0.1787670780733d-02	0.5445973899158d-04
	0.1483989444747d-02	0.5205589054135d-04

node #	solution norm	error norm
	0.2750152231701d-02	0.8581944297319d-04
	0.1891073556826d-02	0.6053136101483d-04
12	0.1929296577680d-02	0.6108192472432d-04
	0.1778218618858d-02	0.5441010065660d-04
	0.1466249589158d-02	0.5190838635941d-04
	0.2740938691518d-02	0.8582891635768d-04
	0.1884676602079d-02	0.6050785242421d-04
13	0.1921008072365d-02	0.6103449663750d-04
	0.1769093979427d-02	0.5436089652194d-04
	0.1449427485525d-02	0.5176151560717d-04
	0.2732011154095d-02	0.8583841378365d-04
	0.1878469674019d-02	0.6048470502510d-04
14	0.1912997662637d-02	0.6098746703098d-04
	0.1760280640154d-02	0.5431212265864d-04
	0.1433465703577d-02	0.5161527874204d-04
	0.2723357744030d-02	0.8584793432105d-04
	0.1872444841498d-02	0.6046191420021d-04
15	0.1905251643552d-02	0.6094083308988d-04
	0.1751763329959d-02	0.5426377515996d-04
	0.1418310371827d-02	0.5146967606394d-04
	0.2714967200669d-02	0.8585747705535d-04
	0.1866594600040d-02	0.6043947539612d-04
16	0.1897757124392d-02	0.6089459198706d-04
	0.1743527665595d-02	0.5421585014244d-04
	0.1403910973096d-02	0.5132470772259d-04
	0.2706828844231d-02	0.8586704108752d-04
	0.1860911844923d-02	0.6041738412259d-04
17	0.1890501975661d-02	0.6084874088521d-04
	0.1735560097977d-02	0.5416834374673d-04
	0.1390220153400d-02	0.5118037372443d-04

7.2 Helical Chain

Verification values for HC, computed by final LU node

class	solution norm	error norm	surface integral
S	0.5218111814670d-03 0.3707865163709d-03 0.3800195018967d-03 0.3397401925927d-03 0.2759004448851d-03	0.2089313120425d-04 0.1476501931249d-04 0.1525298768678d-04 0.1366601055353d-04 0.1177099812324d-04	0.7840627336810d+01
W	0.2044902312107d-01 0.1588666348974d-01 0.1504465470195d-01 0.1309985413018d-01 0.8059897367087d-02	0.1321514192265d-01 0.9789942432742d-02 0.8730718331532d-02 0.7382890378185d-02 0.2473115882853d-02	0.1116573970136d+02
A	0.4568545705333d-01 0.3363154291261d-01 0.3229649387852d-01 0.2835838318897d-01 0.2033297513577d-01	0.1826624824076d-02 0.1336130626708d-02 0.1297209081421d-02 0.1142645851712d-02 0.8878915256529d-03	0.1208035142313d+02
B	0.7585910699148d+00 0.5696105127190d+00 0.5268880230462d+00 0.4574775990656d+00 0.2476122970461d+00	0.3039692982264d-01 0.2267152193556d-01 0.2124188635570d-01 0.1851684149150d-01 0.1138970412935d-01	0.1243337273105E+02

7.3 Visualization Pipeline

Verification values for VP, computed by final FT node

class	*Re*(checksum)	*Im*(checksum)
S	-8.994899992758d+04	-3.251690423310d+04
W	-4.655638928393d+06	-5.590164304487d+04
A	-5.741701238090d+06	-3.237222308450d+04
B	-1.188946561666d+07	-2.079141648557d+05

7.4 Mixed Bag

Verification values for MB, computed by FT nodes in final graph layer

class	col.index	*Re*(checksum)	*Im*(checksum)	# iterations
S	0	-8.977825791271d+04	-3.245251953627d+04	2
	1	-8.963654068307d+04	-3.324636067181d+04	1
	2	-8.935307569342d+04	-3.313613239670d+04	1
W	0	-3.892598836945d+06	-4.672989296527d+04	2
	1	-3.529619667886d+06	-4.606914807530d+04	1
	2	-2.803661281496d+06	-3.656395484244d+04	1
A	0	-5.371673190742d+06	-3.047905369011d+04	2
	1	-5.352454993544d+06	-3.573635637856d+04	1
	2	-5.314018425421d+06	-3.554626087459d+04	1
B	0	-1.087289228420d+07	-9.834694435147d+04	5
	1	-9.935110564550d+06	-1.481787099725d+05	3
	2	-8.055335522632d+06	-1.200930753600d+05	3
	3	-8.051124343134d+06	-1.200284365321d+05	3

References

[1] D. Bailey, E. Barscz, J. Barton, D. Browning. R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, S. Weeratunga. *The NAS Parallel Benchmarks.* NAS Technical Report RNR-94-007, NASA Ames Research Center, Moffett Field, CA, 1994.

[2] D.H. Bailey, T. Harris, W.C. Saphir, R.F. Van der Wijngaart, A.C. Woo, M. Yarrow. *The NAS Parallel Benchmarks 2.0.* NAS Technical Report NAS-95-020, NASA Ames Research Center, Moffett Field, CA, 1995.

[3] M.A. Frumkin, R.F. Van der Wijngaart. *NAS Grid Benchmarks: A Tool for Grid Space Exploration.* Cluster Computing, Vol. 5, No. 3, 2002

[4] R.F. Van der Wijngaart, M.A. Frumkin. *NAS Grid Benchmarks Version 1.0.* NAS Technical Report NAS-02-005, NASA Ames Research Center, Moffet Field, CA, 2002

[5] `http://www.nas.nasa.gov/Research/Software/`